An Object-Oriented Modeling of the History of Optimal Retrievals

Yong ZHANG[1], Vijay V. RAGHAVAN[1] and Jitender S. DEOGUN[2]

[1]Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA 70504–4330

[2]Department of Computer Science and Engineering
University of Nebraska
Lincoln, NE 68588–0115

Abstract

Learning techniques are used in IR to exploit user feedback in order that the system can improve its performance with respect to particular queries. This process involves the construction of an optimal query that best separates the documents known to be relevant from those that are not. Since obtaining relevance judgments and constructing an optimal query involve a great deal of effort, in this paper, we develop a framework for organizing the history of optimal retrievals. The framework involves the identification of a hierarchy of document classes such that the concepts corresponding to higher level classes are more general than those of the lower level classes.

The ways in which such a hierarchy may be used to retrieve answers to new queries are outlined. This approach has the advantage that the query specification is concept-based, where as the retrieval mechanism is numerically-oriented involving optimal query vectors.

It is shown that the construction of a hierarchy of optimal queries can correspond to an object-oriented modeling of IR objects. Furthermore, the resulting model can be easily implemented using a relational DBMS.

1. Introduction

In an Information retrieval (IR) environment, it is impossible to perform perfect retrieval; that is, to precisely select only and all relevant documents [1]. In order to provide reasonable retrieval or optimal retrieval, researchers in IR have introduced learning techniques that exploit user feedback in order to improve retrieval effectiveness [1,2,3]. Most of the earlier work on the use of learning techniques have considered the enhancement of the retrieval results separately for each query instance by obtaining an optimalquery. For the purpose of this work, an *optimal query* is defined to be a decision rule that optimally separates the documents judged relevant by a user from those judged non-relevant among a sample of documnents examined by the user. But, whatever the system learns may be lost at the end of processing that query. It is well known that the learning process is very expensive in the sense that it takes time and effort to obtain the relevance judgements that make up the samples in the training set. For an IR system, the knowledge that the system gains by the learning process relative to some user need might include the optimal query, the set of documents retrieved by this query and some properties about the query. If the knowledge relating to the history of optimum retrievals is accumulated, the effort for learning could be avoided, when users want to repeat the same request as the one in the history. More importantly, when a new information request is close to one in the history, the learning process for obtaining the corresponding optimal query would be shortened and the effort could be lessened by using the knowledge in the history.

For keeping the history of optimum retrieval, the system should support a long-term storage structure for accumulating the results of the learning process over the past. Obviously, the manner in which the system organizes this knowledge critically affects the utility of this effort.

In this paper, we propose a hierarchical structure in which the knowledge about the past optimum retrievals is organized into conceptual entities at different levels of generality. By introducing concepts from the Object-oriented approach, we shall denote the conceptual entities as object classes, around which knowledge such as the description of the class, the operations allowed on the class and the rules that govern the behavior of the class and its operations can be grouped.

We regard documents as objects. A set of documents forms an object class. The optimal queries that retrieve a

set of documents are treated as the operations on this object class and the rules for deciding an optimal query form the rule part of the class. The criterion of forming an object class is based on the conceptual cohesiveness between sets of documents which we shall define later.

When a user has an information requirement in mind, he usually doesn't know the form of the optimal query with respect to his need. However he should have a certain intuitive conceptual idea about his requirement. Starting with that initial knowledge, hopefully, the user can navigate through the historical hierarchy and locate the documents or a related optimal query through which the desired information can be retrieved quickly. This is the motivation of our research in this paper.

The remainder of the paper is organized as follows. In section 2, the notions of conceptual distance and conceptual cohesiveness are defined which follow from the ideas proposed by Kodratoff and his coworkers [4]. However here we specialize the ideas to fit the situations in IR domain. Based on these notions, a hierarchy generating algorithm is presented in section 3. The general flow of the algorithm is derived from that proposed in [4]. But in our context, we desire a hierarchy of object classes instead of a hierarchy of prototypes. From the point of view of object-oriented database design, the automatic construction of a hierarchy of object classes is very important in the data modeling step [5]. Therefore, it is natural to consider the building of such a hierarchy on top of a database management system in order to take advantage of the facilities provided by the DBMS. In section 4, one of the data modeling techniques, called OMT, that is useful in implementing a hierarchy over a relational database is briefly illustrated. In addition, some strategies for making use of the knowledge in the history of optimum retrievals based on the database implementation are presented. Finally, the conclusions of this study are given in section 5.

2. Conceptual distance and cohesiveness

In this section, we first describe an algorithm to learn generalizations from a set of samples. Then the notations of conceptual distance and cohesiveness are defined in order to determine the preferred generalization.

2.1 Concept learning from example

Concept learning from example is a kind of inductive machine learning technique. The learning system is presented with independent instances representing a certain class, and the task is to induce a general description of the class [6].

Considering the result of an optimum retrieval as a potentially good grouping mechanism, the system is presented with a set of relevant documents representing a class. By using the technique of concept learning from example, the learning system obtains a general description of the class that can be regarded as the concept for the class.

Based on the concepts learned from the sets of documents that were optimally retrieved during the past, the system will cluster similar concepts in the sense that they have small conceptual distances between themselves and in turn obtain more general descriptions (concepts) by the technique of concept learning from example. This process is called clustering by generalizing and will generate a hierarchy of concepts.

Intuitively, we would like to measure conceptual distance between two concepts by inspecting how many common properties they both have. The goal of concept learning from example is to induce a preferable generalization of the examples. The generalization is able to reflect the conceptual distance that we intuitively prefer. Subsequently, we could formally define the notions of conceptual distance and cohesiveness based on the generalization.

First, we provide an outline of the learning algorithm we chose to derive the preferable generalization from examples or concepts.

2.2 The learning algorithm for obtaining generalization

In this algorithm, both examples and the generalization are described in the same representation language, which is of the form of conjunctions of literals. All literals have the from:

$$(T, x)$$

where T is a term or keyword and x is the argument of T.

The algorithm is based on the principle of a structural matching. That is, the examples (or concepts) are successively transformed until they acquire approximately the same form. Then the generalization is obtained by retaining only the common features. This kind of learning algorithm has been developed at LRI [7]. But in the realm of IR, some details should be treated differently.

To illustrate this approach, let us consider the following two examples. For the sake of simplicity, the conjunction symbols between literals are omitted. Let documents d_1 and d_2 have the expressions as below:

d_1: (Pattern-Recognition , 3)(Classification , 4)

d_2: (Computer-Vision , 2)(Pattern-Recognition , 4) (Scene-Analysis , 5)

One of the operations of the algorithm is to rewrite the examples (or concept) so as to reveal their common features. For the given examples, we have:

d_1: (Pattern-Recognition , X2)(Classification , 4)

d_2: (Computer-Vision , 2)(Pattern-Recognition ,X2) (Scene-Analysis , 5)

Next, the algorithm will use the theorems of the representation language in order to identify features of one example that imply those exhibited by the other. For instance, documents have term "Pattern-Recognition" should have some importance for term "Computer-Vision". Thus, there is the theorem:

$$\forall_v(\text{Pattern-Recognition}, v) \Rightarrow \exists_x(\text{Computer-Vision}, x)$$

Using this theorem one can rewrite the two examples as follows:

d_1: (Computer-Vision , X1)(Pattern-Recognition , X2) (Classification , 4),

d_2: (Computer-Vision , X1)(Pattern-Recognition , X2) (Scene-Analysis , 5).

When no other common features may be revealed, one simply drops the differing features between the two expressions and obtains a generalization of the initial examples:

$G(d_1,d_2)$: (Computer-Vision , X1)(Pattern-Recognition , X2)

2.3 The contributions of terms to similarity and dissimilarity

The terms of d_1, d_2, and $G(d_1,d_2)$ could be classified in to one of three categories as follows:

• Common terms

Common terms refer to the terms from $G(d_1,d_2)$ which were initially present in d_1 and d_2, such as

$$(\text{Pattern-Recognition}, X2)$$

For the common terms, their contributions to similarity and dissimilarity are defined as follows:

$$S(d_1, d_2, T) = 1$$
$$D(d_1, d_2, T) = 1 - S(d_1, d_2, T) = 0$$

• Dropped terms

Terms deleted from d_1 and d_2 in order to obtain $G(d_1,d_2)$ are called Dropped terms, for example:

(Classification , 4) and (Scene-Analysis , 5)

The dropped term are regarded as an indication of dissimilarity between the two examples. Therefore, the contribution of a dropped term T to the dissimilarity and similarity between d_1 and d_2 are defined as follows:

$$D(d_1, d_2, T) = 1$$
$$S(d_1, d_2, T) = 1 - D(d_1, d_2, T) = 0$$

• Terms introduced by Theorems

Terms in $G(d_1,d_2)$ that are introduced by theorems should be treated as the common terms. Such terms as

(Computer-Vision , X1)

Their contributions to the similarity and dissimilarity are the same as those of the common terms.

2.4 Relative importance of the terms

The process of obtaining a generalization of a set of examples (or concepts) is not a deterministic one. Our objective is to find a good generalization, which is not necessarily the one that considers all properties common to the examples to be equally important, but rather the one that gives different weights depending on the relative importance of properties common to the examples. Generally speaking, one would prefer the generalization that has the desirable property of reflecting the most important commonality between the documents. To achieve this goal, the relative importance of the various terms should be given by a domain expert in the form of a weight associated to each term. In this context, general terms may be regarded as more important than specific ones. For instance, if T is a common term with weight ω, then its contribution to the similarity and dissimilarity will be taken as

$$S(d_1, d_2, T) = 1 * \omega = \omega, \text{ and}$$
$$D(d_1, d_2, T) = 1 - S(d_1, d_2, T) = 1 - \omega$$

If T is one of the dropped terms, then its influence should be

$$D(d_1, d_2, T) = 1 * \omega = \omega, \text{ and}$$
$$S(d_1, d_2, T) = 1 - D(d_1, d_2, T) = 1 - \omega$$

2.5 Conceptual distance and conceptual cohesiveness

The total measure of similarity and dissimilarity estimated for a generalization G are defined respectively as

$$S(d_1, d_2, G) = \sum_{T_i \in \{CM,TH,DR\}} S(d_1, d_2, T_i), \text{ and}$$

$$D(d_1, d_2, G) = \sum_{T_i \in \{CM,TH,DR\}} D(d_1, d_2, T_i),$$

where CM is the set of common terms, TH is the set of terms introduced by theorems, and DR refers the set of dropped terms.

Now let the quality of G be measured by the distance function

$$f(d_1, d_2, G) = D(d_1, d_2, G)/S(d_1, d_2, G)$$

Then the *conceptual distance*, which corresponds to the preferred generalization, is defined as

$$\text{conceptual} - \text{distance}(d1, d2) = \min_{G(d_1,d_2)} \{f(d_1, d_2, G(d_1, d_2))\}$$

The reciprocal of the conceptual distance is called the *conceptual cohesiveness* of d_1 and d_2. That is,

$$\text{conceptual-cohesiveness} = 1/\text{conceptual-distance}$$

Moreover, the generalization for which f is a minimum is considered as the concept to be learned from d_1 and d_2. We denote such a generalization as $G^*(d_1,d_2)$.

The definitions above can easily be extended for any number of examples. Let us consider n documents d_1, d_2, ... , d_n. Let $G(d_1, d_2, \ldots, d_n)$ be one of their generalizations. Based on this generalization, one can compute the similarity $S(d_1, d_2, \ldots, d_n, G)$, dissimilarity $D(d_1, d_2, \ldots, d_n, G)$, and the distance function $f(d_1, d_2, \ldots, d_n, G)$. Then, the conceptual distance is the minimum of f over all possible generalizations of $d_1, d_2, \ldots, d_n$. The reciprocal of the conceptual distance is called the conceptual cohesiveness of the set $\{d_1, d_2, \ldots, d_n\}$. The generalization with minimum f is denoted as $G^*(d_1, d_2, \ldots, d_n)$.

3. Construction of a hierarchy of object classes

In the previous section, we have discussed how to learn a concept corresponding to a set of documents. In this section, we describe an algorithm to construct a hierarchy of the concepts by learning concepts as generalizations of other concepts. Then the hierarchy of the concepts will be transformed to the hierarchy of the object classes. We shall illustrate this process by an example.

3.1 Concepts learning from classes of documents

As a part of the history of optimum retrieval, suppose that the system is presented with many independent classes of documents, say $\{S_1, S_2, \ldots, S_n\}$. By applying the learning algorithm discussed in the preceding section, the system will learn a concept (generalization with minimum f) for each of the classes. The corresponding set of concepts is obtained as $\{C_1, C_2, \ldots, C_n\}$, where $C_i = G^*(S_i)$ for i = 1, 2, ..., n. Every concept is of the form of conjunction of literals.

We shall illustrate the process by an example with a collection of documents shown in Table 1.

With the representation language described in section 2, for example, d_1 and d_{12} can be expressed as:

d_1 = (Pattern-Recognition, 2)(Classification, 4) (Parametric, 4)(Bayes-Rule, 3)

d_{12} = (Classification, 5)(Parametric, 5)(Probabilistic-Model, 1)

Indexing Vocabulary	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12
Computer-Science	0	0	1	0	0	0	0	0	2	2	0	0
Pattern-Recognition	2	3	0	0	0	0	4	3	4	0	0	0
Classification	4	4	0	0	0	0	6	5	3	0	0	5
Parametric	4	0	0	4	0	0	0	0	0	0	5	5
Bayes-Rule	3	0	0	0	0	0	0	0	0	0	0	0
Non-parametric	0	3	0	0	5	0	0	0	0	0	0	0
Information-Retrieval	0	0	4	3	2	2	0	0	0	3	0	0
Boolean-Model	0	0	5	0	0	3	0	0	0	0	0	0
Probabilistic-Model	0	0	0	4	5	0	0	0	0	0	4	1

Table 1. A collection of documents

Suppose that the history of optimum retrievals contains six optimal queries and sets of relevant documents retrieved by the queries.

Q_1: $\{d_1, d_{12}\}$

Q_2: $\{d_1, d_2, d_7, d_8, d_9\}$

Q_3: $\{d_2\}$
Q_4: $\{d_3, d_5, d_6, d_{10}\}$
Q_5: $\{d_3, d_6\}$
Q_6: $\{d_4, d_{11}\}$

For the learning system, suppose it has the theorems listed as below:

1. $\forall_v$(Pattern-Recognition, v) $\Rightarrow$
   $\exists_x$(Computer-Science, x)

2. $\forall_v$(Information-Retrieval, v) $\Rightarrow$
   $\exists_x$(Computer-Science, x)

3. $\forall_v$(Probabilistic-Model, v) $\Rightarrow$
   $\exists_x$(Information-Retrieval, x)

4. $\forall_v$(Classification, v) $\Rightarrow \exists_x$(Pattern-Recognition, x)

5. $\forall_v$(Bayes-Rule,v) $\Rightarrow \exists_x$(Probabilistic-Model,x)

The relative importances of the terms given by an expert are as shown in Table 2.

Table 2: Relative importances of terms

Term Name	Weight
Computer-Science	1.0
Pattern-Recognition	0.8
Information-Retrieval	0.8
Classification	0.7
Boolean-Model	0.4
Probabilistic-Model	0.4
Parametric	0.4
Non-Parametric	0.4
Bayes-Rule	0.3

Then, by using the learning algorithm described in Section 2, we may get two different generalizations for the document set $\{d_1, d_{12}\}$ retrieved by Q_1. They are as shown below:

$G_1(d_1, d_{12})$ = (Classification,X1)(Parametric, X2)
(Probabilistic − Model, X3)

with

CM_1 = {Classification, Parametric};
TH_1 = {Probabilistic-Model};
DR_1 = {Pattern − Recognition, Bayes-Rule};

$$S(d_1, d_{12}, G_1) = 0.7 + 0.4 + 0.4 + (1 - 0.8) + (1 - 0.3) = 2.4$$
$$D(d_1, d_{12}, G_1) = (1 - 0.7) + (1 - 0.4) + (1 - 0.4) + 0.8 + 0.3 = 2.6$$
$$f(d_1, d_{12}, G_1) = 2.6/2.4 = 1.08$$

and

$G_2(d_1, d_{12})$ =(Pattern − Recognitiio, X1)
(Classification,X2)(Parametric, X3)

with

CM_2 = {Classification, Parametric};
TH_2 = {Pattern-Recognition};
DR_2 = { Bayes-Rule, Probabilistic − Model};

$$S(d_1, d_{12}, G_2) = 0.7 + 0.4 + 0.8 + (1 - 0.3) + (1 - 0.4) = 3.2$$
$$D(d_1, d_{12}, G_2) = (1 - 0.7) + (1 - 0.4) + (1 - 0.8) + 0.3 + 0.4 = 1.8$$
$$f(d_1, d_{12}, G_2) = 1.8/3.2 = 0.56$$

Conceptual-distance$(d_1, d_{12}) = \min\{1.08, 0.56\} = 0.56$
Conceptual-cohesiveness=1/0.56=1.79

Therefore, $G_2(d_1,d_{12})$ is considered as the concept learned from the document set $\{d_1,d_{12}\}$. We denote it as C_1.

Omitting the details of the process, the concepts learned from the remaining retrieved

sets of relevant documents are listed below:

C_1 = (Pattern-Recognition,X1)(Classification,X2) (Parametric,X3)

C_2 = (Computer-Science,X1)(Pattern-Recognition,X2) (Classification,X3)

C_3 = (Pattern-Recognition,3)(Classification,4) (Non-parametric,3)

C_4 = (Computer-Science,X1)(Information-Retrieval,X2)

C_5 = (Computer-Science,X1)(Information-Retrieval,X2) (Boolean-Model,X3)

C_6 = (Information-retrieval,X1)(Probabilistic-Model,X2) (Parametric,X3)

Where C_i represents the concept learned from the set of relevant documents retrieved by Q_i, for i = 1, 2, ..., 6.

3.2 The clustering algorithm

Given the set of concepts $\{C_1, C_2, ..., C_n\}$, the clustering algorithm will cluster the concepts by generalizing in such a way that the conceptual cohesiveness of each cluster is maximized.

Given a set of concepts $\{C_1, C_2, ..., C_n\}$, the algorithm first looks for the two concepts C_i, C_j for which the conceptual cohesiveness is maximum. These concepts form the seed of a cluster. A new concept C_k is added to this cluster only if the conceptual cohesiveness of the set $\{C_i, C_j, C_k\}$ does not decrease very much. Therefore, a threshold should be given in order to be able to determine when the conceptual cohesiveness has decreased significantly.

The threshold is denoted by the symbol μ, where $0 < \mu \leq 1$. Based on μ , the following statements are established:

1. $c(S_1) \ll c(S_2) \Longleftrightarrow c(S_1) < \mu * c(S_2)$
2. $c(S_2) \ll c(S_1) \Longleftrightarrow c(S_2) < \mu * c(S_1)$
3. $c(S_1) \equiv c(S_2) \Longleftrightarrow \neg c(S_1) \ll c(S_2) \& \neg c(S_2) \ll c(S_1)$

where $c(S_i)$ denotes as the conceptual cohesiveness of the set S_i. For a particular application, one should experimentally determine the value of the threshold.

The algorithm, which is extended from the one proposed by Kodratoff *et al*[4], is described below. Assume that the threshold μ has been determined.

Step 1: Compute the conceptual cohesiveness of each pair of concepts

Let $C=\{C_1, C_2, ..., C_n\}$ be the set of concepts. For each pair $\{C_p, C_q\}$, compute its conceptual cohesiveness defined previously.

Step 2: Choose a seed of the clustering.

Determine the pair $\{C_p, C_q\}$ for which $c(C_p, C_q)$ is maximum. Let $M=\{C_p, C_q\}$.

Step 3: Determine the concepts which could be members of the cluster represented by the chosen seed

That is, determine the set:

$$T = \{C_k | c(C_p, C_q) \equiv c(C_k, C_p) \& c(C_p, C_q) \equiv c(C_k, C_q)\}$$

If $c(C_k, C_p) \ll c(C_p, C_q)$,
then $c(C_p, C_q, C_k) \ll c(C_p, C_q)$. Therefore C_k may not be the member of the cluster represented by $\{C_p, C_q\}$.

Step 4: Introduce concepts into the cluster

For each C_k from T, if $c(C_p, C_q) \equiv c(M, C_k)$, then introduce C_k into M.

Here, if $M = \{C_p, C_q, ..., C_t\}$, then $c(M, C_k)$ means $c(C_p, C_q, ..., C_t, C_k)$

At the end of this step one has discovered the cluster represented by the seed $\{C_p, C_q\}$:

$$M = \{C_p, C_q, ..., C_s\}$$

and also obtained the generalization (concept) of C_p, C_q,..., C_s as C_M=G(C_p, C_q,..., C_s) for which the f is minimum.

Step 5: Replace the concepts contained in the cluster M with the concept C_M

Remove elements of the discovered cluster M from the set of concept C. That is:

$$C \leftarrow (C - M) \cup \{C_M\}$$

Step 6: Rerun the algorithm

Repeat from step1 with the new set of concepts until C can not be further reduced.

This clustering algorithm is able to discover a hierarchy of concepts characterizing sets of documents and computes a description of each concept in the form of generalization.

For the given example, the algorithm is presented with a set of concepts $C=\{C_1, C_2, ..., C_6\}$, and the resolution is determined as μ=0.8. Running the clustering algorithm with the set of concepts, we obtained the hierarchy of concepts shown in Figure 1.

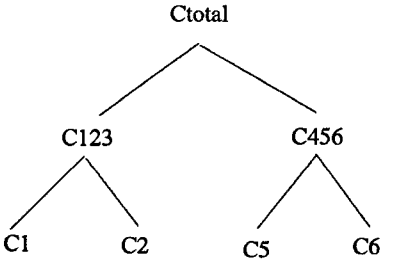


Figure 1. The hierarchy of concepts

In the Figure 1,

C_{total} = G*(d1, d2, ..., d12) = (Computer-Science, X1)

C_{123} = G*(d1, d2, d7, d8, d9, d12)
= (Computer-Science, X1)(Pattern-Recognition, X2) (Classification, X3)

C_{456} = G*(d3, d4, d5, d6, d10, d11)
= (Computer-Science, X1)(Information-Retrieval, X2)

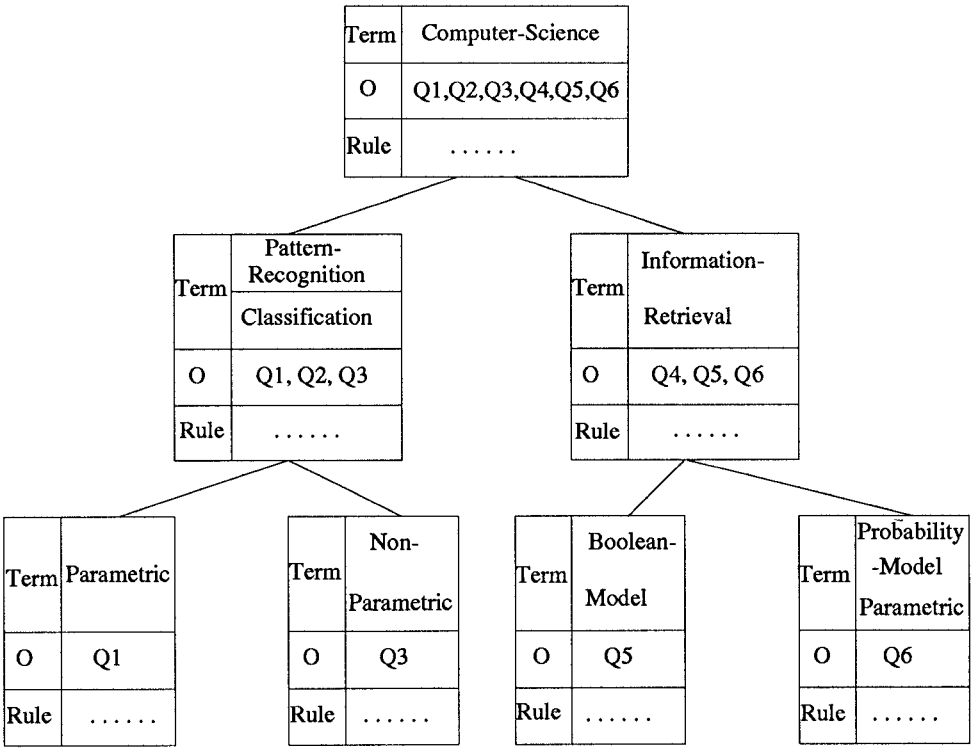


Figure 2. A hierarchy of object classes

3.3 A hierarchy of object classes

Objects and object classes are the primary components in the paradigm of object-oriented technology [8]. An object class represents a set of objects that represent a concept in the real world. Usually, an object class consists of three parts; description part, operation part and rule part. The description part contains attributes of the class. The operation part exhibits the possible operations that are allowed on the object class. The rule part is the knowledge base in which rules for the operations are specified.

A hierarchy of object classes means that the object classes are ordered in a class-subclass hierarchy in which each object class inherits all the properties of its superclass. Thus, the knowledge about objects and object classes is organized within the hierarchy in a memory efficient manner.

In the domain of information Retrieval , an object refers to a document. Therefore, an object class represents a set of documents. Aside from sets of documents, the history of optimal retrievals includes the optimal queries and some additional knowledge about these queries, such as the restrictions for applying the optimal queries and the characteristics of the queries.

As a result of the clustering algorithm, we discover many clusters of concepts. Since each of the concepts represents one or several sets of documents which were retrieved by one or several optimum queries, we can treat the clusters as object classes. The terms appearing in the description of a cluster are the attributes of the object class. The corresponding optimal queries are the operations of the object class. The knowledge about optimal queries forms the rule part of the object class.

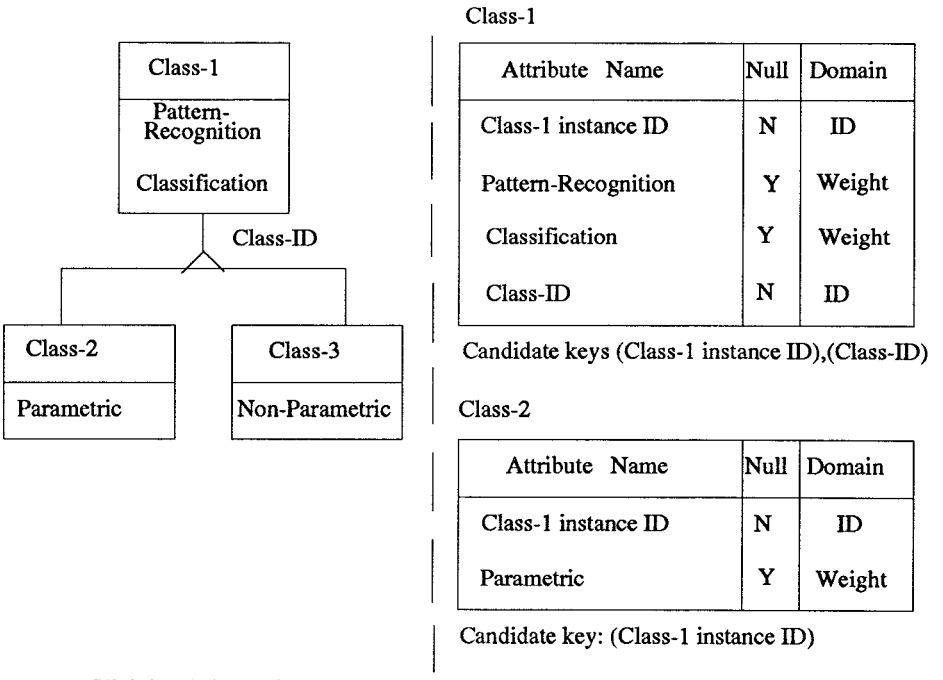
The clustering algorithm is used to generate a hierarchy of concepts. For transforming it to a hierarchy of object classes, the only step that remains is to compute the description specific to each object class (node in the tree). This can be done by removing, from the description of each object class, the terms which are already present the descriptions of its ancestors.

After instantiating the operation and the rule parts of each object class respectively with the corresponding optimum queries and the additional knowledge about the queries, we finish the task of constructing a hierarchical structure for the history of optimum retrievals.

For the example given earlier, we obtain the hierarchy of object classes shown in Figure 2, which is transformed from the hierarchy of concepts shown in Figure 1.

4. Building the hierarchy in a DBMS

The hierarchy of object classes can be implemented in a relational DBMS in order to benefit from the facilities it provides and the theoretical basis provided by the relational model. We briefly show an implementing technique and describe some possible strategies for the application



(a) High level abstraction

(b) The idael relation tables

Figure 3 Representations for Generalization

of the history of optimal retrievals on the basis of that implementation.

4.1 Data Modeling

A data model is the first design step towards using a database in an application. It defines the structure of a database [5]. Furthermore, the object-oriented approach can be employed to define useful abstractions on top of a standard data model. When the underlying data model is relational, object-oriented data modeling promotes adherence to normal forms and improves integration between a database and its applications[11].

Generalization is an extremely valuable abstraction mechanism supported by object-oriented modeling technique. A hierarchy of object classes based on generalization allows the inheritance of the properties of the object classes in the upper levels of a hierarchy by the classes that are below. In the proceeding sections, we discussed the construction of the hierarchy of object classes by generalizing. This hierarchy can be considered as a high level abstraction of the database model. Such an abstraction can be implemented by an actual DBMS.

OMT (Object Modeling Technique) is a relational database design approach based on object-oriented concept [11]. By OMT, the hierarchy of object classes based on generalization we discussed previously can be transformed into relational database schema.

Figure 3(b) gives the result of transforming the generalization hierarchy of object classes a (high level) shown in Figure 3(a) into ideal relational tables. The triangle in Figure 3(a) symbolizes generalization. In Figure 3(b), we do not show the table of Class-3 since you may obtain it by replacing the attribute "Parametric" in the table of Class-2 by the attribute "Non-Parametric".

Although, in the representation of OMT, the operation and rule parts are not exhibited, they are implicitly associated with each table as secondary information. In addition, the process of transformation from high level abstraction to a target relational database schema could be automatically carried out.

4.2 Strategies for the application of the history

By building the history of optimum retrievals on relational database, one could benefit a lot from the many facilities provided by a DBMS and from the strong theoretical basis that underlies the relational model. Of course, for our application, some extensions to the SQL of relational database would be desirable [12]. We don't discuss this issue in this paper. What we are interested in are the strategies for retrieving the information present in the history of optimum retrievals, assuming that the necessary extensions have been done.

Since the idea is to illustrate the concept involved in the strategies rather than to show the relational language

explicitly, we explain the retrievals with natural language. Several strategies are listed below.

- Boolean Retrieval

By Boolean retrieval we mean the use of queries in which the usual Boolean operators are applied to specify index term combinations.

Operation 1a: Find the document ID's for all documents indexed by "Pattern-Recognition","Classification" and "Non-parametric", but not by "Linear-discriminant".

But, just retrieving the documents by Boolean retrieval does not fully exploit the knowledge in the history of optimum retrievals. When one's information requirement can be met by processing an optimal query in the history, the following operations will usually take place.

Operation 1b: Create a table for all documents indexed by "Pattern-Recognition", "Classification" and "Non-Parametric", but not by "Linear-discriminant".

(This operation finds the most general object class which has the desirable attributes.)

Operation 1c: Find the optimal queries associated with the object class.

Operation 1d: Select one of the optimal queries according to some knowledge.

Operation 1e: Apply the optimal query to the documents contained in the created table to obtain optimum result.

If none of the optimum queries in the history could obtain a satisfactory result, one could select an optimal query that is the most promising. Then, the usual learning process of constructing an optimum retrieval would be performed by taking this optimal query as the initial query. By following this strategy, the learning process may be expected to converge faster than the process of learning the optimal query from scratch.

- Weighted Boolean Retrieval

Here we are interested in retrieving information that satisfies several criteria. For instance:

Operation 2: Find the optimal queries associated with the documents which have weight of "Pattern-Recognition" larger than 3, and weight of "image-Processing" less than 2.

When users have more accurate information about their request, this kind of retrieval could be used.

- Weighted Retrieval

In fact, the operation 1e is a form of weighted retrieval. In order to perform this retrieval, we assume the existence of a special function, SIM( ), appropriate to the given query language that computes the similarity between a query and a given document. For instance,

Operation 3: List and rank the IDs of documents that have similarities higher than 0.60 with the query Q.

- Feedback Searches

Feedback searches are those which involve the results of a previous search. For example, a user may examine the results of a search and decide that he prefers certain documents. For example, suppose that documents numbered 25 and 167 are relevant, but the document numbered 2501 is not. Two kinds of operations, which are illustrated below, may be performed.

Operation 4a: List the ID's of documents that are like documents 25 and 167 but not like document 2501.

For this operation, the system will presents the documents in the object class of maximum size that contains documents 25 and 167 but not 2501. As discussed previously, documents in a class have high conceptual cohesiveness between them.

Operation 4b: Find an optimal query that retrieves documents 25 and 167 but not document 2501.

Since system has some knowledge about optimal queries that are contained as a part of object classes, the system could determine a more appropriate optimal query based on the characteristics of the optimal queries and the properties that documents 25, 167 and 2501 have.

Further strategies to make use of information provided by the history of optimum retrieval could be given. However, most needs can be met by the strategies already shown. The object-oriented modeling technique enriches the flexibility of the system with respect to both search strategies and information representation. The semantic linkages are specified at definition time. Therefore, the structure of the query language is simple and the operations shown above can be expressed in a straightforward manner [14].

5. Conclusion

Optimal retrieval may be based on numerical approaches. It could provide more reasonable retrieval results with respect to user's information requirements than the conceptual retrieval, such as Boolean retrieval, which involves a symbolic approach [1]. But the former approach suffers from the following major disadvantage: the conceptual meaning is not explicitly expressed by the optimal queries. Thus, it is hard for users to obtain and manage them. The symbolic approach, in contrast, has the advantage that it could explicitly express the users' concepts in terms of some keywords (primitive concepts). Such kind

of queries can be well understood and can be appealing to the intuition of human beings [17].

Several models for IR that include features of both symbolic and numerical methods [15,16] have been investigated. Most of them focus on improving retrieval performance by converting a symbolic query into a numerical expression. But these approaches are unable to achieve performance that is as good as the pure numerical approach.

Therefore, in the work of this paper, our attention is focused on a different scheme for combining the features of both symbolic and numerical approaches. In order to retain good retrieval performance, the retrieval operations are performed by optimal queries that are obtained by some learning technique. Then, by using the machine learning approach, the symbolic concepts relevant to the optimal query can be learned from the set of objects retrieved by the query. Based on the concepts generated, a conceptual clustering method is applied to construct a hierarchy which organizes the results of optimal retrievals around meaningful conceptual entities. Thus, user's information requirement can be first expressed by a symbolic query. Based on the concepts contained in the query, a corresponding optimal query, which will perform optimum retrieval, can be selected.

In this model, the performance of document retrieval depends solely on the method that one adopts to learn the optimal queries, rather than on the conceptual hierarchy constructed. However, the problem of how to locate an optimal query that is most appropriate to meet a particular user's requirement is still open. We will investigate this problem in our future work.

Reference

1. Nobert Fuhr. Optimum Polynomial Retrieval Functions Based on the Probability Ranking Principle. *ACM Trans. on Information Systems,* Vol. 7, No. 3, July 1989. pp 183–204.

2. V. V. Raghavan , J. S. Deogun and P. Rhee. Formulation of the Term Refinement Problem for User-oriented Information Retrieval. *Proceedings of The Annual AI Systems in Government Conference,* Washington, D.C., March 1989, pp 72–78.

3. S. K. M. Wong, Y. Y. Yao and P. Bollman. Linear structure in information Retrieval. *ACM 11th International Conference on Research & Development in Information Retrieval,* Grenoble-France, 1988, pp 219–232.

4. Y. Kodratoff and G. Tecuci. Learning based on Conceptual Distance. *IEEE Trans. on Pattern Analysis and Machine Intelligence.* Vol 10, No. 6, Nov. 1988, pp 897–909.

5. R. Hull, R. King. Semantic Database Modeling: Survey, applications, and Research issues. *ACM Computing Surveys,*19(3), Sept. 1987, pp 201–260.

6. R. S. Michalski, J. G. Carbonell, and T.M. Mitchell, Eds., *Machine Learning: An Artificial Intelligence Approach,* Vol 2. Palo Alto, CA: Morgan —Kaufman, 1986

7. Y. Kodratoff, and J. G. Ganasciua. Improving the generalization step in learning, In *Machine Learning: An Artificial Intelligence Approach,* Vol. 2, 1986, pp. 215–244.

8.A. Goldberg, and D. Robson. *Smalltalk-80: The language and Its Implementation.* Addison-Wesley, Reading, Mass. 1984.

9. V. V. Raghavan and G. S. Jung. Connectionist Learning in Constructing Thesaurus-like Knowledge Structure. *AAAI Symposium on Text-Based Intelligent Systems,* Spring, 1990, pp 123–127.

10. J. S. Deogun and V. V. Raghavan. User-oriented Document Clustering: A Framework for Learning in Information retrieval, *Proc. Ninth Annul International ACM Conf. on Research and Development* in Information retrieval, Pisa-Italy, 1986.pp. 157–163.

11. M. R. Blaha, W. J. Premerlani, and, J. E. Rumbaugh. Relational Database Design using an Object-Oriented Methodology. *Communications of the ACM.* Vol. 31, No. 4, April 1988, pp 414–427.

12. D. D. Chamberlin, R. F. Boyce, SEQUEL: A structured English Query Language. *Proceedings of the ACM-SIGMOD Workshop on Data Description,* Access, and Control. Ann Arbor MI, 1974, pp 249–264.

13. R. G. Crawford. The Relational Model in Information Retrieval. *Journal of the American Society for Information Science.* ,1981, pp 51–63.

14. I. A. Macleod, and A. R. Reuber. The Array Model: A Conceptual Modeling Approach to Document Retrieval. *Journal of the American Society for Information Science.* 38(3) 162–170, 1987.

15. G. Salton, E. A. Fox, and H. Wu. Extended Boolean Retrieval. *Communications of the ACM,* 26: 1022–1036, 1983.

16. S. K. M. Wong, W. Ziarko, V. V. Raghavan, P. C. N. Wong. On Modeling of Information Retrieval Concepts in Vector Spaces. *ACM trans. on Database Systems,* Vol. 12, No. 2, June 1987, pp 299–321.

17. C. T. Yu, W. Meng, and S. Park. A Framework for Effective Retrieval. *ACM Transactions on Database Systems,* 14(2), June 1989, pp147–167.